参考：<http://blog.csdn.net/lizhitao/article/details/51718156>

**腾讯 VS 阿里 VS 携程消息中间件设计方案及思路**

原创 2016年06月20日 01:28:45

* 6217

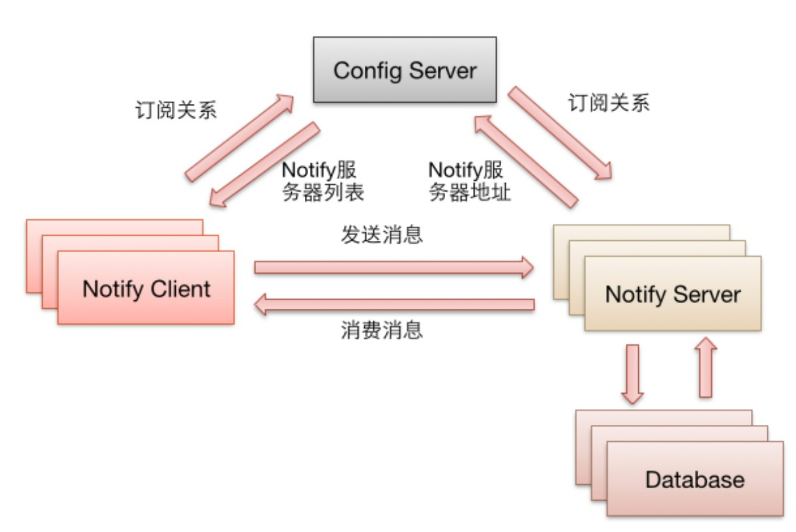
背景

目前我们美团正在设计和不断迭代、升级消息中间件方案，为了避免走弯路，希望站在巨头肩膀上，学习经验、吸取精华，推动美团MQ快速演进，为美团业务高速扩张提供支撑

目标：可靠性（保证消息不丢失）、异步、解耦（无需同时在线、不需要知道对方是谁）。   
数据的存储级别：内存中的数据（断电丢数据）===》持久化磁盘（磁盘损坏）===》冗余备份（一致性问题）

业界MQ设计方案如下：

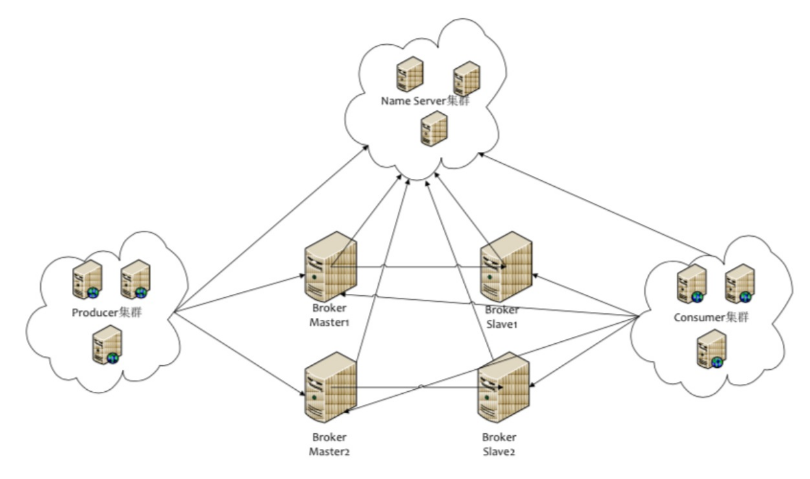
1.阿里Notify架构



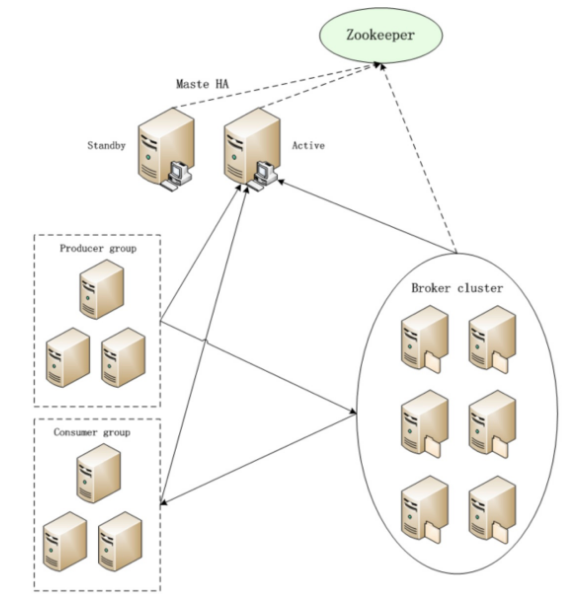
特点：

* Notify之间不互相通讯。
* 支持水平扩展。
* 客户端通过Config Server获得Notify地址列表。
* 客户端自动感知Notify的增加或减少。
* 发布者、消费者、Notify Server都支持集群。
* 消息根据不同的安全级别选择存放到不同的地方（如：File、Oracle、Mysql），然后放在内存中提高性能。
* 推模式

2.阿里RocketMQ架构

两主两从部署模式：   


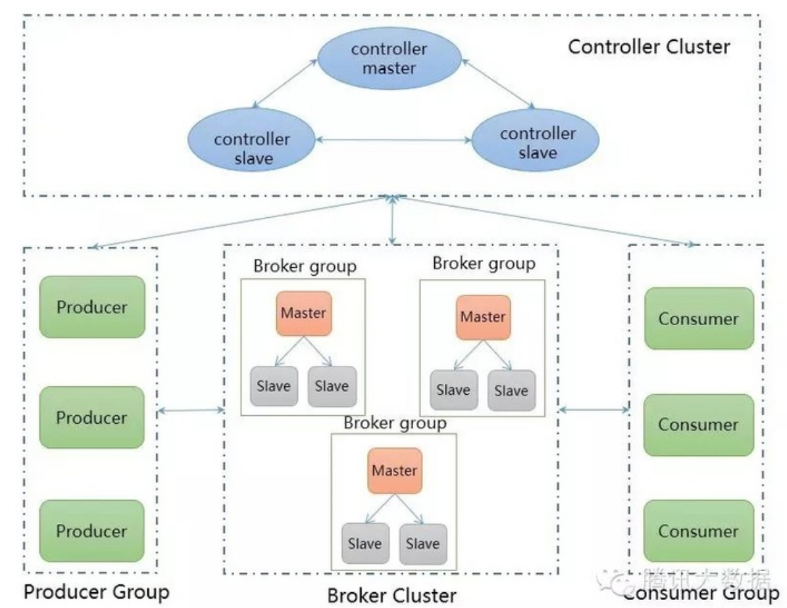
特点：

* Name Server无状态节点，节点之间无任何信息同步。
* Broker与Name Server集群中的所有节点建立长连接，定时注册Topic信息到所有Name Server。
* Producer与Name Server中的其中一个节点建立长连接，定期获取Topic路由信息。
* Consumer与Name Server中的其中一个节点建立长连接，定期从NameServer获取Topic路由信息。
* 拉模式   
  3.腾讯-Tube架构   
  

特点：

* Tube集群使用了Zookeeper，目前主要用来保存Consumer的消费位置和Master HA的选举（历史遗留问题，全新的Tube系统设计可以摆脱对ZK的依赖）
* Broker向Master汇报自身信息，包括自身id、状态以及提供哪些Topic的发布和订阅服务，每个Topic下包含多少分区。
* 生产者和消费者向Master通报topic信息，返回从哪些Broker获取数据（客户端自己做负载均衡）
* Broker集群节点之间通过心跳和Master保持状态同步，当状态发生变化时，Master会负责通知相关节点。
* Master采用主备模式，通过ZK来进行选举。
* 拉模式

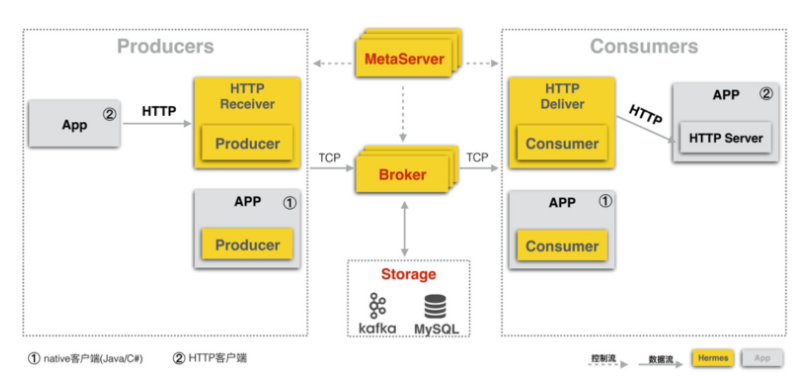
4.腾讯-Hippo架构



特点：

* 三台controller 一主两备承担整个系统节点数据的采集。（主备controller于心跳检测，在主故障的时候自动failover）
* 三台broker一主两备组成一个组，主broker向controller定期汇报心跳以告知controller当前组的存活状态。（主备broker之间存在心跳，主broker挂掉后，重新选举，shuffle）
* producer与controller之间存在心跳，获取topic所在组的broker组的ip端口机器queue信息。
* consumer与controller之间存在心跳，获取broker组信息列表+同组其他消费者信息列表。
* 限时锁定：消费者拉取某个队列的数据与确认回调之间设置一个超时时间，一旦超时时间还没确定，自动解锁。
* 提供控制台界面，根据当前收集到的正常运行的broker节点信息，可以指定给某个特定的broker组下发topic及queue添加事件。
* 拉模式

5.携程-Herms架构

  
\* broker加入/退出，consumer加入/退出，parition的负载均衡。   
\* metaserver通过zk发现broker，自己创建路由表，并分配给broker。   
\* broker定时向meta server定时续lease。（通过zk做协调）   
\* consumer尽量不直接连接zk，consumer到meta server获取lease。（考虑到consumer量大，不通过zk做协调，直接和metaserver竞争lease）   
\* 可通过meta server做直接干预（如机器出现问题）。   
\* 长轮询pull模式，早期使用推模式，broker需要写的代码很复杂，而且一些高级特性不方便支持。

引用来自涂扬整理wiki